A Unified Flow Scheduling Method for Time Sensitive Networks

Mingwu Yao, Jiamu Liu, Jing Du, Dongqi Yan, Yanxi Zhang, Wei Liu and Anthony Man-Cho So[†]

School of Telecommunications Engineering, Xidian University, Xi'an, 710071, Shaanxi, China
†Department of Systems Engineering and Engineering Management, Chinese University of Hongkong, Hongkong SAR, China

ARTICLE INFO

Keywords:
TSN
Scheduling
Network Utilization
Remaining Time
Unified Framework
Mixed Initial Population
Genetic Algorithm

ABSTRACT

Given the network and the time-triggered flow requests of a Time Sensitive Network (TSN), configuring the gate control lists (GCL) of IEEE 802.1Qbv for the ports of each node can be formed as a Job Shop Scheduling Problem, which is NP-hard. At present, most of the existing heuristic solutions for such problems consider scenarios where all given traffic flows can be scheduled. In order to solve the undetermined flow scheduling problem in scenarios no matter whether the flows can be scheduled or not, we propose to maximize the remaining time in conjunction with optimizing the network utilization instead of only minimizing the flowspan. Though the new problem is still NP-hard, it is a unified framework capable of covering general scenarios. On the basis of the new framework, we propose a novel Mixed initial population Genetic Algorithm (MGA) to solve the problem. Extensive simulation evaluation shows that MGA performs better and faster in different network scenarios while other methods prevails only in specific scenarios. This feature makes the method attractive in realistic TSN scheduling applications for in most cases it is hard for users to properly classifying the problem.

1. Introduction

As guaranteed real-time communication[1] is important to many applications, such as Industry 4.0, Smart Grids, and Internet of Things, the demands for ultra-low latency communication (ULLC) are significantly growing[2, 3]. One such effort to meet the requirement is IEEE standard of Time Sensitive Network (TSN). By TSN, critical traffic flows with bounded low latency and jitter will share the channel with other flows, thus improving efficiency and reducing cost[4].

The TSN supports different classes of traffic, such as prioritized scheduled traffic and best-effort (BE) traffic[1]. The IEEE 802.1Qbv proposed by TSN Task Group defines enhancements for the scheduled traffic[5]. In TSN, prioritized traffic should be carried by end-to-end, periodic scheduled flows (SFs), of which the frames are periodically forwarded along a non-cyclic path. In each node through the path, the sending of SF frames in each outgoing port is controlled by the opening and closing of gates according to the open and close time schedules configured in the gate control lists (GCLs)[5]. The GCLs will be periodically executed on a network-wide time basis established by time synchronization protocols. If properly set, each packets of an SF will pass through each node along its path in a TSN right when the corresponding gate is open[5].

Given the TSN network topology and all the SF requests for the network, the calculation of all the time configurations of GCLs is a scheduling problem consisting of two sub-problems, i.e., a start-time-calculating problem and a sequencing problem[6]. Finding an acceptable sequential order of the SFs, by start-time-calculating, the GCLs can be determined. Therefore, solving the scheduling problem equals to finding the optimum ordering of the given SF sets. The problem is equivalent to a bin-packing problem and is therefore NP-hard[7].

*Corresponding author: MW Yao
ORCID(s):

The scheduling of SFs can be categorized into static [2][6] and dynamic ones[8][9]. For TSN, currently, most of the scheduling methods of SFs are static, such as those deployed in manufacturing applications[10].

The scheduling problem can be formulated into integer linear programming and solved by tools, e.g., CPLEX[2] and SMT solvers[8]. Due to computation burdens, however, heuristic algorithms are preferred[11, 12, 13, 14], such as Tabu search[6], Greedy Randomized Adaptive Search Procedure[12], and genetic algorithms[13][15].

Once the SFs are scheduled, the remaining bandwidth can be used by lower priority traffics, such as BE traffic. Typically, a guard band is inserted between the closing time of lower traffic gates and the opening time of scheduled traffic gates to isolate them. However, too many guard bands will reduce the remaining bandwidth for lower traffics and scheduling with fewer guards bands is preferred[6].

For a given topology and SFs, if it is known that all the SFs can be scheduled for any order, it is named as determined scenario. Otherwise it is undetermined. As the problem is NP-hard, it is often impossible to assume a priori to which scenario a specific problem belongs. However, most of the algorithms mentioned above are applicable to the determined one and may not be valid, or at least be efficient, for the undetermined, implicitly or explicitly, due to the difficulties to include different optimization goals. The method proposed for the undetermined scenario is only to schedule more flows[14], which ignores the individual differences between flows. Therefore, it is more desirable to cover the undetermined scenarios with a unified framework, which naturally includes the determined.

In this paper, we investigate the undetermined scheduling problem of TSN, aiming at maximizing network utilization while keeping available bandwidth for the lower traffic as much as possible. And without loss of generality, we assume that there is only scheduled traffic and BE traffic in the network and the path of each SF is given as the shortest.

The main contributions of this work are twofold:

1) A unified framework to schedule the SFs in more general scenarios: Using hop delay, the new model includes both Store-and-Forward switches (SFX) and Cut-Through switches (CTX). With the remaining time, the framework can avoid the drastic variation of the evaluation criteria during scheduling in undetermined scenarios.

2) MGA (the Mixed initial population Genetic Algorithm): Designed on the basis of the unified framework, MGA can produce better GCLs to increase the available bandwidth for BE traffic while maximizing the network utilization in many different TSN scenarios.

The rest of this paper is organized as follows: The problem and the unified framework are described in Section II. Section III presents the MGA. The evaluation results produced by extensive simulations are discussed in section IV and the last section presents conclusions.

Notations: We use calligraphic letters, e.g., $\mathcal{E}$,$\mathcal{F}$, to denote sets, uppercase bold letters, e.g., $\mathbf{A}$ and $\mathbf{D}$ for matrices and lowercase bold letters, e.g., $\mathbf{d}$ and $\mathbf{p}$, for vectors. $*$ is used as a wild card to denote any possible element of its domain.

2. System Model and Problem Statement

2.1. System Model

We consider a TSN network supporting the IEEE 802.1Qbv gate control protocol. The network is comprised of switches (SW), end systems (ES), and Ethernet links[16], as shown in Fig. 1, and there are many scheduled traffic need to be carried by the network. We are expected to find some feasible scheduling of these high priority traffics according to their transmission periods and frame lengths, while leaving as many bandwidth as possible to BE traffic [6]. The frames of a specific SF are generated periodically by its source and transmitted hop by hop along a given non-cyclic path. Both SW and ES have ingress and egress ports connected via Ethernet links[3]. As shown in Fig. 2, of each egress port the frames waiting to be forwarded on the associated link waiting in queues[17], each equipped with a gate[18]. The frames in a queue are allowed to be transmitted only if the corresponding gate is open, one at a time[15]. A GCL associated with each port contains an ordered list of gate operations. In this paper, one queue is used for scheduled traffic, while others are for BE traffic. With GCLs, the scheduled traffic can be transmitted in the reserved time slices to meet its stringent latency and jitter requirements, and ensure it is not interfered by BE traffic. For instance, in Fig. 2, at time instance T_{02} only the scheduled traffic gate is open, while at time instance T_{30} all the gates are closed. The GCL repeats itself with a time cycle T_{cycle} upon a basis of network-wide time synchronization.

For a network with N vertices of SWs and ESes, and M simplex Ethernet links, the network topology can be modeled as a directed graph $G(\mathcal{V},\mathcal{E})$[2], where $\mathcal{V} = \{0,1,2,...,\mathrm{N}-1\} \subset \mathbb{N}$, $\mathcal{E} = \{0,1,2,...,\mathrm{M}-1\} \subset \mathbb{N}$. $v \in \mathcal{V}$ is corresponding to SW and ES, and the directed edges

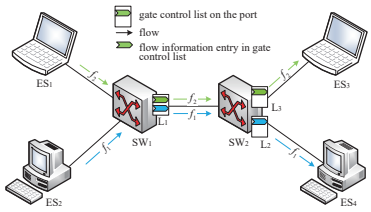


Figure 1: A TSN network of two switches (SW_1 and SW_2), four end systems (ES_1, ES_2, ES_3 and ES_4), and two SFs (f_1, f_2). As paths f_1 and f_2 have a common link the configuration of all GCLs(L_1, L_2, L_3) should avoid conflicts.

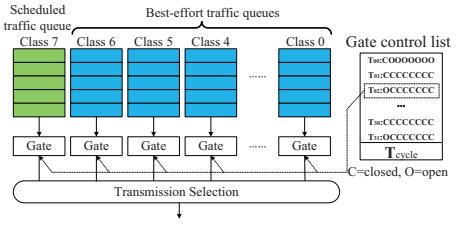


Figure 2: A Qbv-capable port architecture with GCL[5]

Table 1
Scheduled Flow Parameters

parameters	meaning
$\mathbf{f}_F \in \mathcal{E}^{\lvert\mathcal{F}\rvert}$	vector of the first edge of flows: $\mathbf{f}_F[f] = e$, iff the first hop of the flow path $f \in \mathcal{F}$ is edge $e \in \mathcal{E}$
$\mathbf{l}_F \in \mathcal{E}^{\lvert\mathcal{F}\rvert}$	vector of the last edge of flows: $\mathbf{l}_F[f] = e$, iff the last hop of flow path $f \in \mathcal{F}$ is edge $e \in \mathcal{E}$
$\mathbf{p}_F \in \mathbb{N}^{\lvert\mathcal{F}\rvert}$	vector of flow period: $\mathbf{p}_F[f] = p_f$, iff flow $f \in \mathcal{F}$ has a period of $p_f \in \mathbb{N}$

$e \in \mathcal{E}$ the simplex Ethernet link. An edge e_i which departs at vertex v_d and enters vertex v_e is denoted as $e_i = (v_d, v_e)$. And the adjacency matrix $\mathbf{A}_{EE} \in \{0,1\}^{|\mathcal{E}|\times|\mathcal{E}|}$ is defined as

$$\mathbf{A}_{EE}[e_p][e_n] = \begin{cases} 1, & \text{if } e_p = (*, v) \text{ and } e_n = (v, *) \\ 0, & \text{otherwise} \end{cases}. \quad (1)$$

For the two edges $e_p, e_n \in \mathcal{E}$, only when e_p ends at a vertex $v \in \mathcal{V}$ and e_n starts at the same vertex v, $\mathbf{A}_{EE}[e_p][e_n] = 1$.

The definition of the relationship matrix between edges and vertices, $\mathbf{B}_{VE} \in \{-1,0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$, is defined as

$$\mathbf{B}_{VE}[v][e] = \begin{cases} 1, & \text{if } e = (v, *) \\ -1, & \text{if } e = (*, v) \\ 0, & \text{otherwise} \end{cases}. \quad (2)$$

When the starting vertex of the edge e is v, $\mathbf{B}_{VE}[v][e] = 1$, and when the ending vertex of e is v, $\mathbf{B}_{VE}[v][e] = -1$.

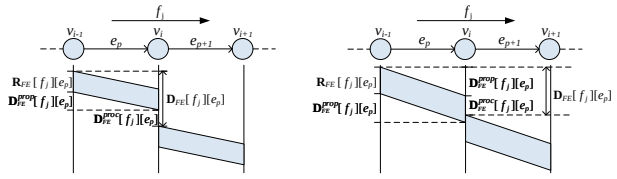
We denote the set of all given K SFs in the network as $\mathcal{F} = \{0,1,2,..,\mathrm{K}-1\} \subset \mathbb{N}$. And some flow parameters in the network are defined in Table 1.

The flow-edge relationship matrix $\mathbf{U}_{FE} \in \{0,1\}^{|\mathcal{F}|\times|\mathcal{E}|}$ is

$$\mathbf{U}_{FE}[f][e] = \begin{cases} 1, & \text{if flow } f \text{ passes edge } e \\ 0, & \text{otherwise} \end{cases}. \quad (3)$$

Of the delays we discussed, the transmission delay is the time a frame takes to leave the output queue, i.e., the bit-wise frame-length divided by the link rate. The propagation delay is the time each bit passing the link. The latency from the input port to the output queue is the processing delay. And the queuing delay is the time a frame waiting in the output queue [19]. In TSN it is preferred to schedule SFs tightly properly to completely eliminate the queuing delay.

The matrix of transmission delay is defined as $\mathbf{R}_{FE} \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{E}|}$, of which $\mathbf{R}_{FE}\left[f_j\right]\left[e_p\right]$ is that of flow f_j on edge e_p. And we denote the propagation delay matrix as $\mathbf{D}_{FE}^{prop} \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{E}|}$, of which $\mathbf{D}_{FE}^{prop}\left[f_j\right]\left[e_p\right]$ is that of f_j on e_p. Then we denote the matrix of processing delay as $\mathbf{D}_{FE}^{proc} \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{E}|}$, of which $\mathbf{D}_{FE}^{proc}\left[f_j\right]\left[e_p\right]$ is that of f_j on the end vertex of e_p.



(a) Store-and-Forward (b) Cut-Through

Figure 3: Delays in different types of switches

The hop delay is defined as the latency between two departing events of a frame from two consecutive egress ports along its path. As one hop corresponds to one edge, we denote the matrix of hop delays of flows over edges as $\mathbf{D}_{FE} \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{E}|}$. We denote the jitter of the network as j. When $\mathbf{U}_{FE}\left[f_j\right]\left[e_p\right] = 1$, the $\mathbf{D}_{FE}\left[f_j\right]\left[e_p\right]$ is the hop delay of the transmission of f_j on e_p. As shown in Fig. 3, $\mathbf{D}_{FE}\left[f_j\right]\left[e_p\right] = \mathbf{D}_{FE}^{proc}\left[f_j\right]\left[e_p\right] + \mathbf{D}_{FE}^{prop}\left[f_j\right]\left[e_p\right] + \mathbf{R}_{FE}\left[f_j\right]\left[e_p\right] + j$, for the SFX. $\mathbf{D}_{FE}\left[f_j\right]\left[e_p\right] = \mathbf{D}_{FE}^{proc}\left[f_j\right]\left[e_p\right] + \mathbf{D}_{FE}^{prop}\left[f_j\right]\left[e_p\right] + j$, for the CTX [20]. When $\mathbf{U}_{FE}\left[f_j\right]\left[e_p\right] = 0$, let $\mathbf{D}_{FE}\left[f_j\right]\left[e_p\right] = -1$ for computation convenience.

2.2. The Flow Scheduling Problem

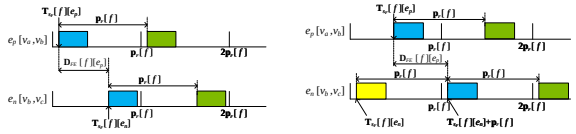
For a given flow set and a network, we need to find a configuration of the GCLs of all the ports, so that all the SF data frames immediately start transmission at the very moment of arriving without queuing delay, known as no-queuing configuration[6]. In this case, when an SF is scheduled, we assume that its end-to-end delay bound is satisfied.

the frames of each SF are generated periodically, and different flows may have different period. The least common multiple of all flow periods, h, is defined as the hyper-cycle[2]. And T_{cycle} of all the GCLs is usually set as the hyper-cycle.

When the topology and flow information, i.e., path, period and delays, are given, the configuration for one SF in all the GCLs along its path is determined as long as the start time of the flow on its source is determined. If all the start times of all SFs are obtained, the overall GCL configurations are achieved. We denote the start time vector of the SFs on their corresponding sources as $\mathbf{s}_F \in \mathbb{N}^{|\mathcal{F}|}$. When $\mathbf{s}_F[f_i] = -1$, it means f_i will not be scheduled. When $0 \le \mathbf{s}_F[f_i] < \mathbf{p}_F\left[f_i\right]$, $\mathbf{s}_F[f_i]$ is the start time of f_i on its source from the beginning of the hyper-cycle, which is intrinsically less than the period of the flow, $\mathbf{p}_F\left[f_i\right]$.

Given the start time vector $\mathbf{s}_F$, the start time matrix is defined as $\mathbf{T}_{s_F} \in \mathbb{N}^{|\mathcal{F}|\times|\mathcal{E}|}$, of which the element $\mathbf{T}_{s_F}[f][e]$ is the start time of flow f on e along its path, if $\mathbf{U}_{FE}[f][e] = 1$ and $\mathbf{s}_F[f] \neq -1$. $\mathbf{T}_{s_F}[f][e] = -1$ otherwise.

When e is the first edge of f, $\mathbf{T}_{s_F}[f][e] = \mathbf{s}_F[f]$. Recursively, given $\mathbf{T}_{s_F}[f]\left[e_p\right]$ of a previous edge e_p, the starting time of the next edge e_n along the path can be calculated as $\mathbf{T}_{s_F}[f]\left[e_n\right] = \left(\mathbf{T}_{s_F}[f]\left[e_p\right] + \mathbf{D}_{FE}[f]\left[e_p\right]\right) \bmod \mathbf{p}_F[f]$. The modular operation is for both the cases shown in Fig. 4, where in Fig. 4(b) the transmissions of one frame in e_p and e_n are not in the same flow period in the two neighboring GCLs.



(a) Non period crossing case (b) Period crossing case

Figure 4: The Gantt chart of two different frame transmissions

When f_i can be scheduled, the transmission duration of each frame of f_i is called as reserved time slice. With $\mathbf{T}_{s_F}$, we can calculate all the reserved time slices in the network.

For two transmissions on the same edge, if the first transmission is finished earlier than the start of another, then the two will not conflict. As $\mathbf{T}_{s_F}\left[f_i\right][e] + a \cdot \mathbf{p}_F\left[f_i\right]$ is the start time of the $(a+1)$th transmission of f_i on e, and $\mathbf{T}_{s_F}\left[f_i\right][e] + b \cdot \mathbf{p}_F\left[f_i\right] + \mathbf{R}_{FE}\left[f_i\right][e]$ is the finish time of the $(b+1)$th transmission of f_i on e, when the two transmissions meet either (4) or (5), they will not conflict.

It is required that the reversed time slice of any period of any SFs do not overlap, so all periods in a hyper-cycle should be considered, as shown in (6) and (7). Since the frames are sent periodically and some reserved time slices may shift to the next period of f, there will be $\dfrac{h}{\mathbf{p}_F[f]}$ or $\left(\dfrac{h}{\mathbf{p}_F[f]} + 1\right)$ frames of the same flow to be considered in a hyper-cycle.

The scheduling constraint is that all the reserved time slices of the configuration for any link do not conflict in the hyper-cycle. That is, $\forall f_1, f_2 \in \mathcal{F}, f_1 \neq f_2, \forall e \in \mathcal{E}, \forall a \in A$, and $\forall b \in B$, if $\mathbf{U}_{FE}\left[f_1\right][e] + \mathbf{U}_{FE}\left[f_2\right][e] = 2$, $\mathbf{s}_F[f_1] \neq -1$ and $\mathbf{s}_F[f_2] \neq -1$) then:

$$\mathbf{T}_{s_F}\left[f_1\right][e] + a\mathbf{p}_F\left[f_1\right] \ge \mathbf{T}_{s_F}\left[f_2\right][e] + b\mathbf{p}_F\left[f_2\right] + \mathbf{R}_{FE}\left[f_2\right][e] \tag{4}$$

$$\text{or } \mathbf{T}_{s_F}\left[f_2\right][e] + b\mathbf{p}_F\left[f_2\right] \ge \mathbf{T}_{s_F}\left[f_1\right][e] + a\mathbf{p}_F\left[f_1\right] + \mathbf{R}_{FE}\left[f_1\right][e]$$

$$\text{with } \mathcal{A} = \left\{ a \in \mathbb{N}\colon 0 \le a \le \frac{h}{\mathbf{p}_F\left[f_1\right]} \right\} \tag{6}$$

$$\text{and } \mathcal{B} = \left\{ b \in \mathbb{N}\colon 0 \le b \le \frac{h}{\mathbf{p}_F\left[f_2\right]} \right\}. \tag{7}$$

We define the set of $\mathbf{s}_F$ of all the no-queuing configurations satisfying the scheduling constraint as $\mathcal{S}_{FV} \subset \mathbb{N}^{|\mathcal{F}|}$, which is the solution space of the scheduling problem.

As in [21], the link utilization of edge e is defined as the ratio of the sum of all the reserved time slices on e to the time length of the hyper-cycle. For all the links in the network, we denote the vector of link utilization as $\mathbf{lu}_{s_F} \in \mathbb{Q}^{|\mathcal{E}|}$, for any given $\mathbf{s}_F \in \mathcal{S}_{FV}$, where $\mathbf{lu}_{s_F}\left[e_p\right]$ is the link utilization of e_p and h is hyper-cycle length, as shown in (8).

$$\begin{aligned}\mathbf{lu}_{s_F}\left[e_p\right] &= \sum_{\substack{f\in\mathcal{F}\colon \mathbf{s}_F[f]\neq -1,\\ \mathbf{U}_{FE}[f]\left[e_p\right]=1}} \left[\frac{\mathbf{R}_{FE}\left[f\right]\left[e_p\right] \cdot \frac{h}{\mathbf{p}_F[f]}}{h} \right] \\ &= \sum_{\substack{f\in\mathcal{F}\colon \mathbf{s}_F[f]\neq -1,\\ \mathbf{U}_{FE}[f]\left[e_p\right]=1}} \left[\frac{\mathbf{R}_{FE}\left[f\right]\left[e_p\right]}{\mathbf{p}_F\left[f\right]} \right] \end{aligned} \tag{8}$$

Furthermore, the network utilization, lu_{s_F}, is defined as

$$lu_{s_F} = \frac{\sum_{e_p\in\mathcal{E}} \mathbf{lu}_{s_F}\left[e_p\right]}{|\mathcal{E}|}. \tag{9}$$

With network utilization defined, it is also desirable to reduce the number of guard bands by scheduling SFs consecutively compact[6].

2.3. Remaining Time

In this section we introduce a new metric, remaining time, to evaluate the compactness of the scheduling of SFs.

Firstly we review an existing metric, flowspan[6]. For the flow f_i, its flowspan equals to the start time of the last reserved time slice in its first hop plus the end-to-end delay of f_i. We denote the end-to-end delay vector for SFs as $\mathbf{d}_F^{E2E} \in \mathbb{N}^{|\mathcal{F}|}$ where $\mathbf{d}_F^{E2E}[f_i]$ is the end-to-end delay of flow f_i. And we denote the vector of flowspan as $\mathbf{fs}_{s_F} \in \mathbb{N}^{|\mathcal{F}|}$, for the given $\mathbf{s}_F \in \mathcal{S}_{FV}$. If $\mathbf{s}_F[f_i] = -1$, we define $\mathbf{fs}_{s_F}[f_i] = -1$, else if $\mathbf{s}_F[f_i] \neq -1$, $\mathbf{fs}_{s_F}[f_i]$ is the flowspan of flow f_i:

$$\begin{aligned}\mathbf{fs}_{s_F}\left[f_i\right] &= \left(\frac{h}{\mathbf{p}_F\left[f_i\right]} - 1\right) \cdot \mathbf{p}_F\left[f_i\right] + \mathbf{s}_F[f_i] + \mathbf{d}_F^{E2E}[f_i] \\ &= h - \mathbf{p}_F\left[f_i\right] + \mathbf{s}_F[f_i] + \mathbf{R}_F\left[f_i\right]\left[\mathbf{l}_F\left[f_i\right]\right] \\ &+ \sum_{e_p\in\mathcal{E}:\mathbf{U}_{FE}[f_i][e]=1\&e\neq\mathbf{l}_F[f_i]} \mathbf{D}_F\left[f_i\right][e]. \end{aligned} \tag{10}$$

For the given $\mathbf{s}_F \in \mathcal{S}_{FV}$, the maximum of all $\mathbf{fs}_{s_F}[f]$, i.e., $fs_{s_F}^{max} = \max_{f\in\mathcal{F}\wedge\mathbf{s}_F[f]\neq-1} \mathbf{fs}_{s_F}[f]$, is defined as the flowspan of the network. By minimizing $fs_{s_F}^{max}$, the compact schedules could be achieved[6].

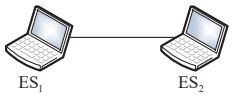


Figure 5: The example point-to-point network with $|\mathcal{V}| = 2$.

Table 2
Example Scheduled Flows Information

flow	origin	destination	period
f_1	ES_1	ES_2	1
f_2	ES_1	ES_2	2
f_3	ES_1	ES_2	4

However, $fs_{s_F}^{max}$ may change drastically when the hyper-cycle changes during the scheduling where the SF set under consideration varies, especially in undetermined scenarios.

In order to overcome the shortcomings of flowspan, we define the remaining time, which is the hyper-cycle minus the flowspan. Similar to flowspan, remaining time is capable of reflecting the quality of the scheduling results, and at the same time, it avoids drastic variation when the hyper-cycle changes during the scheduling process.

For each $\mathbf{s}_F \in \mathcal{S}_{FV}$, the remaining time vector is denoted as $\mathbf{rt}_{s_F} \in \mathbb{N}^{|\mathcal{F}|}$, of which $\mathbf{rt}_{s_F}[f_i]$ is the remaining time of flow f_i if $\mathbf{s}_F[f] \neq -1$. If $\mathbf{s}_F[f_i] = -1$, let $\mathbf{rt}_{s_F}[f_i] = -1$.

$$\begin{aligned}\mathbf{rt}_{s_F}\left[f_i\right] &= h - \mathbf{fs}_{s_F}\left[f_i\right] = \mathbf{p}_F\left[f_i\right] - \mathbf{s}_F[f_i] \\ &- \mathbf{R}_F\left[f_i\right]\left[\mathbf{l}_F\left[f_i\right]\right] - \sum_{\substack{e_p\in\mathcal{E}:\mathbf{U}_{FE}[f_i][e]=1\\ \&e\neq\mathbf{l}_F[f_i]}} \mathbf{D}_F\left[f_i\right][e]. \end{aligned}$$

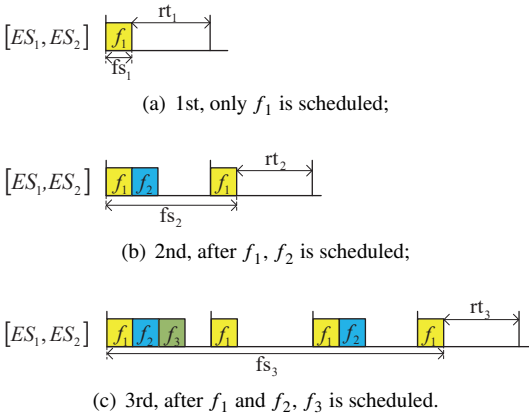
Note that the remaining time could be negative.

The network remaining time (NRT) is defined as the minimal remaining time of all successfully scheduled SFs, i.e., $rt_{s_F}^{min} = \min_{f\in\mathcal{F}\wedge\mathbf{s}_F[f]\neq-1} \mathbf{rt}_{s_F}[f]$. Maximizing $rt_{s_F}^{min}$ will result in the most compact schedule.

In contrast to flowspan, $\mathbf{rt}_{s_F}[f]$ will not change with hyper-cycle changing for the same $\mathbf{s}_F[f]$, as (11) shows. The merit can be further illustrated by an example of scheduling for a toy network of two end systems shown in Fig. 5.

In case 1, three SFs, as given in Table 2, with the same transmission delay of 0.25 is to be scheduled in the order of f_1, f_2 and f_3 as shown in Fig. 6. When f_1 is scheduled, the current hyper-cycle h is 1. When consider f_2, h becomes 2. And h turns into 4 when scheduling f_3. As is clearly shown in Fig. 7, during the process the NRT keeps unchanged while the flowspan varies greatly.

In case 2, for the same example network, consider two SF sets: $\mathcal{F}_1$ contains f_1 and f_2, while $\mathcal{F}_2$ contains f_1 and f_3. All the SFs have the same transmission delay of 0.75. We assume that both in $\mathcal{F}_1$ and $\mathcal{F}_2$, only f_1 can be scheduled. As shown in Fig. 8, the results of two sets are exactly the same, although the hyper-cycles are different. The flowspans of them vary, while the remaining times of them keep unchanged.

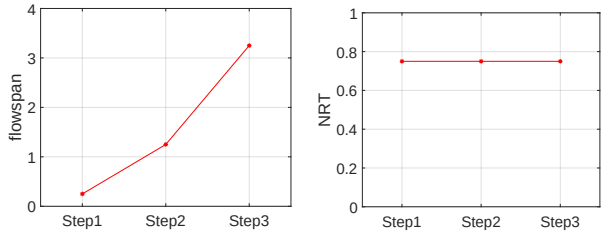


(a) 1st, only f_1 is scheduled;

(b) 2nd, after f_1, f_2 is scheduled;

(c) 3rd, after f_1 and f_2, f_3 is scheduled.

Figure 6: The Gantt charts of scheduling steps of the example, with flowspan(fs) and remaining time(rt) also shown



(a) Flowspan (b) Remaining time

Figure 7: Comparison between the trends of flowspan and remaining time

(a) The scheduling result of $\mathcal{F}_1$ (b) The scheduling result of $\mathcal{F}_2$

Figure 8: The scheduling results of case 2.

The two cases mentioned above clearly show that remaining time is a better evaluation factor than flowspan.

2.4. The Unified Framework

With hop delay, SFX and CTX are readily incorporated in one model. Here a unified framework is proposed using both $rt_{s_F}^{min}$ and lu_{s_F}, to cover both determined and undetermined scenarios.

Firstly we should note there may be more than one $\mathbf{s}_F$ to achieve the maximum NU. We denote the set of them as:

$$\mathcal{T}_{FV} \subset \mathcal{S}_{FV} \& \mathcal{T}_{FV} = \{\mathbf{s}_F^* | \mathbf{s}_F^* = \arg\max_{\mathbf{s}_F \in \mathcal{S}_{FV}} lu_{s_F}\}. \tag{11}$$

Secondly, these NU optimal start time vectors may result in different NRTs. Then we need to find one optimal start time vector $\mathbf{s}_F^{op}$ with maximum NRT in $\mathcal{T}_{FV}$, i.e.,

$$\mathbf{s}_F^{op} = \arg\max_{\mathbf{s}_F^* \in \mathcal{T}_{FV}} rt_{s_F^*}^{min}. \tag{12}$$

To formulate our problem as Integer Linear Program(ILP), we denote flow state vector $\mathbf{y} \in \{0,1\}^{|\mathcal{F}|}$. If flow f_i is scheduled, $\mathbf{y}[f_i] = 1$. If flow f_i is not scheduled, $\mathbf{y}[f_i] = 1$. And the ILP is as follows:

$$\max\left(M \sum_{f \in \mathcal{F}} \left(\mathbf{y}[f] \cdot \sum_{\mathbf{U}_{FE}[f][e_p]=1} \frac{\mathbf{R}_{FE}[f][e_p]}{\mathbf{p}_F[f] \cdot |\mathcal{E}|}\right) - fs_{s_F}^{max}\right) \tag{13}$$

subject to :

$$\forall f \in \mathcal{F}, \mathbf{s}_F[f] \neq -1:$$

$$fs_{s_F}^{max} \geq \mathbf{fs}_{s_F}[f] - M \cdot (1 - \mathbf{y}[f]) \tag{14}$$

$$\forall f_1 \in \mathcal{F}, f_2 \in \mathcal{F}, f_1 \neq f_2, \forall e \in \mathcal{E}, \forall a \in A, b \in B:$$
$$\text{if } (\mathbf{U}_{FE}[f_1][e] + \mathbf{U}_{FE}[f_2][e] = 2 \& \mathbf{y}[f_1] + \mathbf{y}[f_2] = 2) \text{ then:} \tag{15}$$

$$(\mathbf{T}_{s_F}[f_1][e] + a \cdot \mathbf{p}_F[f_1] - \mathbf{T}_{s_F}[f_2][e] - b \cdot \mathbf{p}_F[f_2] - \mathbf{R}_{FE}[f_2][e] \geq -M \cdot x_{f_1,f_2,e,a,b} - M \cdot (1 - \mathbf{y}[f_1]) - M \cdot (1 - \mathbf{y}[f_2]) \tag{16}$$

$$\mathbf{T}_{s_F}[f_2][e] + b \cdot \mathbf{p}_F[f_2] - \mathbf{T}_{s_F}[f_1][e] - a \cdot \mathbf{p}_F[f_1] - \mathbf{R}_{FE}[f_1][e] \geq -M \cdot (1 - x_{f_1,f_2,e,a,b}) - M \cdot (1 - \mathbf{y}[f_1]) - M \cdot (1 - \mathbf{y}[f_2])) \tag{17}$$

$$\text{with } \mathcal{A} = \left\{ a \in \mathbb{N}: 0 \leq a \leq \frac{h}{\mathbf{p}_F[f_1]} \right\} \tag{18}$$

$$\text{and } \mathcal{B} = \left\{ b \in \mathbb{N}: 0 \leq b \leq \frac{h}{\mathbf{p}_F[f_2]} \right\} \tag{19}$$

The constraints of (16) and (17) correspond to (4) and (5.) To this end, we denote $x_{f_1,f_2,e,a,b} \in \{0,1\}$ for each conflict. c is a large constant. According to the value of $x_{f_1,f_2,e,a,b}$, one of (16) and (17) is always true.

If all the given SFs can be scheduled, the scheduling problem has been mapped into No-wait Job Shop Scheduling Problem (NW-JSP), a sub-problem of JSP[6]. As JSP is NP-hard, NW-JSP is also NP-hard [7]. The unified framework consider the undetermined scenarios where some SFs may not be scheduled, therefore, it is NP-hard as well. As our evaluations showing later, it may take too much time to solve the problem using available ILP solvers. We need to look for a heuristic algorithm to solve the problem.Based on the unified framework, a Mixed initial population Genetic Algorithm (MGA) is designed.

3. MGA Flow Scheduling

Given a network and a set of SFs with predefined paths, the SFs can be scheduled sequentially as a permutation of them[6]. We denote the flow sequence vector of a permutation of all SFs as $\mathbf{a}_F \in \mathcal{F}^{|\mathcal{F}|}$, of which $\mathbf{a}_F[i] = f_j$ means f_j is the ith flow to be scheduled. Given an $\mathbf{a}_F$ with the topology

information tI ($\mathcal{E}$, $\mathcal{V}$, $\mathbf{A}_{EE}$ and $\mathbf{B}_{VE}$), and the flow information fI ($\mathcal{F}$, $\mathbf{f}_F$, $\mathbf{l}_F$, $\mathbf{p}_F$, $\mathbf{U}_{FE}$, $\mathbf{R}_{FE}$ and $\mathbf{D}_{FE}$), the $\mathbf{s}_F$ can be directly calculated by Sequence-Based Greedy Scheduling Algorithm (SGSA), which is shown in Algorithm 1. And the worst-case complexity of the algorithm is $\mathcal{O}(N^3M^3)$[6][7], where M is the number of SFs and N is the maximum of the number of SF transmissions in one hyper-cycle.

Algorithm 1 Sequence-Based Greedy Scheduling Algorithm:

```
function SGSA(tI, fI, a_F)
    FStartTime ← {}
    for each f ∈ a_F do
        if f scheduled successfully then
            FStartTime[f] ← Earliest start time of f
        else
            FStartTime[f] ← −1
        end if
    end for
    return FStartTime
end function
```

Genetic algorithms (GA) are used in previous works as quick searching algorithms to find an $\mathbf{a}_F$ yielding good results [13][15]. However, the search space is so huge that the previous works of GA [15] are not satisfactory. The initial population of GA has significant influences on both the quality of the results and the convergence speed[23]. The proposed MGA is featured by its initial population generation and is capable of achieving higher NU and larger NRT.

To solve the scheduling problem with GA, each SF is mapped to one gene, and $\mathbf{a}_F$ is regarded as a genome[13]. Each genome will produce an individual, i.e., a GCL configuration. When two individuals are compared, the one with higher NU is thought better (and survives). If their NU equals, the individual with a larger NRT is thought better.

3.1. The Generation of Initial Population

In some scenarios, scheduling with flow period and hop preference sequence (PHS) yields better results[17]. In PHS, flows are scheduled in the ascending order of their periods. As a flow with smaller periods repeating more often, it is believed more difficult to schedule. When their periods are the same, the one with longer paths will be scheduled first[17].

However, if the flow periods are a little randomized, a random sequence (RS) sometimes prevails PHS, as a toy network given in Fig. 9 shows. It consists of three end systems (ES_1, ES_2, ES_3), two switches (SW_1, SW_2), and three SFs to be scheduled, as shown in Table 3. And the scheduling results of PHS and RS is shown in Fig. 10.

It is natural to conjecture mixing different sequences to generate a better initial population for GA. Considering the impact of hops, we mix PHS, RS and HPS(hop and period preference sequence) to generate initial population. The population generation is shown in Fig. 11. In HPS, flows are scheduled in the descending order of hops. When their hops are the same, the one with smaller period will be scheduled first.

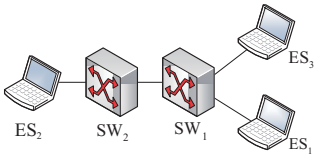
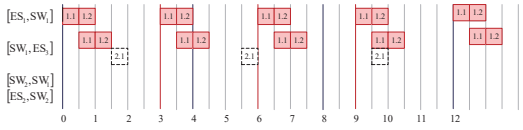


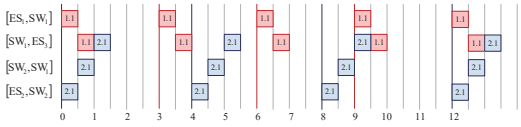
Figure 9: A toy example network with $|\mathcal{V}| = 5$

Table 3
Example Scheduled Flow Information

flow	origin	destination	period	$\mathbf{R}_{FE}[f][*]$
$f_{1.1}$	ES_1	ES_3	3	0.5
$f_{1.2}$	ES_1	ES_3	3	0.5
$f_{2.1}$	ES_2	ES_3	4	0.5



(a) For PHS, NU is 1/6 and $f_{2.1}$ cannot be scheduled



(b) For RS, $f_{1.2}$ cannot be scheduled and NU is 17/96

Figure 10: The Gantt chart of PHS and RS scheduling, neglecting the hop delays. A rectangle (i.j) denotes the transmitted frame of $f_{i,j}$. Each row shows the repetitive occurrence of the frames onto each link. In the example the NU of RS is greater than that of PHS.

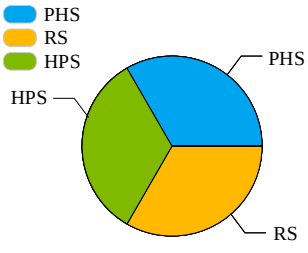


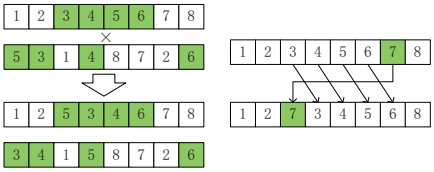
Figure 11: The generation of the initial population of MGA

3.2. The Selection, Crossover and Mutation

Given the initial, the descendant populations will be produced iteratively. In each iteration, the genomes of the selected individuals are crossed and mutated to produce descendants. Our algorithm uses the selection of tournament[24] and elitist[25], combined with Subtour Exchange Crossover (SEC)[26] and Insertion Mutation (IM)[27].

3.2.1. Crossover

As shown in Fig. 12(a), using SEC, a segment of genes is randomly chosen from the genome of parent 1, and each of the genes is found in parent 2. And then the genes are exchanged between the two parents, generating two new individuals[26]. Compared with others, SEC is more suitable to our scheduling problem, as it can readily control the length of crossover genome segments, and does not frequently change the front part of the genome.



(a) SEC (b) IM

Figure 12: Subtour Exchange Crossover (SEC) and Insertion Mutation (IM)

3.2.2. Mutation

With the IM, a gene is randomly taken out from a genome and then inserted back randomly[27], as shown in Fig. 12(b), and only the mutation yielding better results will be retained.

Algorithm 2 MGA:

```
function MGA(tI, fI, s, n)
    curGeneration ← 0
    curPopu ← initPopulation(tI, fI, s)
    while curGeneration < n do
        curPopu ← genePopulation(curPopu, s)
        curGeneration ← curGeneration + 1
    end while
    return opSolution in curPopu
end function
```

Finally, as shown in Algorithm 3, input the topology information tI, flow information fI, population size s, and the limit of generations n, the proposed MGA will output $opSolution$ with the desired start time vector.

4. Performance Evaluation

In this section, MGA is compared with other GAs and ILP solvers in different scenarios. As shown in Fig. 13, a fully connected backbone network is produced by the Erdös–Rényi (ER) random graph model [28], and the access SWs with ESes are added to each node. With a produced network, for each ES as a source, several SFs with random destinations from other ESes are generated, of which the paths are set as the shortest. Their periods are randomly chosen from a predefined integer set. In the evaluation, several different scenarios are generated from the same topology by controlling the number of flows and the integer period set.

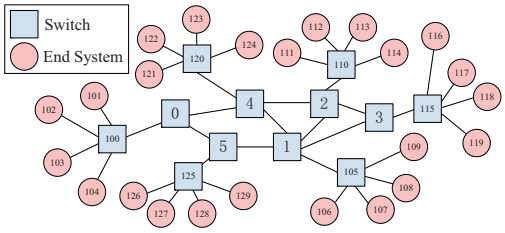


Figure 13: The example of a network topology with $|\mathcal{V}| = 42$

Table 4
ILP Test Scenarios with period = {2ms, 4ms}

scenarios	SW	ES	the number of flows	the number of conflicts
s0	3	6	9	314
s1	9	18	38	2078
s2	8	16	41	2802
s3	9	18	52	3927

Table 5
Results of ILP solution and MGA

scenarios	ILP-Solution			MGA		
	LU	flowspan	time	LU	flowspan	time
s0	0.39	3170000	0.06	0.39	3170000	39.90
s1	0.21	3530000	2.91	0.21	3530000	68.80
s2	0.30	3710000	118.46	0.30	3710000	74.44
s3	0.31	3950000	300.19	0.31	3950000	128.58

4.1. Compared with ILP solver

In scenarios of Table 4, ILP solution and MGA is compared. The number of conflicts is directly related to the dimension of ILP solving matrix.

ILP matrix is generated by preprocessing, and then CPLEX is used to solve the ILP matrix. In this section, we configure the upper time-bound of CPLEX as 300s. The execution time and results of the ILP solution and MGA is shown in Table 5.

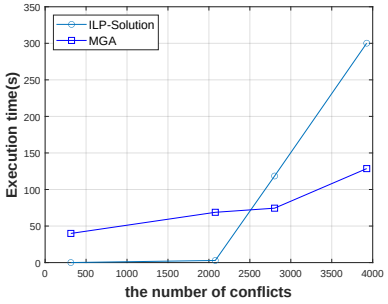


Figure 14: The execution time of ILP and MGA

When there are fewer conflicts in the network, ILP solutions can solve the problem quickly. However, with the complexity of the network growing, there are more possible conflicts in the network, and the execution time of the ILP

Table 6
Test Scenarios

scenarios	period of flows(ms)						the number of flows
	2	3	4	5	6	8	
s4	✓		✓			✓	1936
s5	✓		✓	✓		✓	1125
s6		✓	✓		✓	✓	1733
s7		✓	✓	✓			912
s8	✓		✓			✓	912
s9		✓	✓		✓	✓	469
s10		✓	✓	✓			339

solution increases so fast to be much longer than MGA.

4.2. Compared with other GAs

Then we compare MGA with other GAs using several different population initialization methods in various scenarios of Table 6, of which RGA (Random purebred population GA) adopts RSes as initial populations, while the initial population of PHGA (Period and Hop preference purebred population GA) is all PHSes. And the initial population of HPGA (Hop and Period preference purebred population GA) is all HPSes.

The regular population size is set as 50, and the required number of GA generations is 20. The mutation probability is 0.15. As a contrast, 1000 RSes, 1000 PHSes and 1000HPSes are generated for each scenario.

In scenario s4, s5, s6 and s7, not all of the SFs can be scheduled, where NU is the main comparison object. While s8, s9 and s10 are of comparatively light load, i.e., all the SFs can usually be scheduled, of which NRT is the decisive comparison object, under the premise of maximizing NU. In s4 and s8, the periods of the SFs constitute a geometric sequence. In comparison, the SF periods in s7 and s10 are coprime, consequently, the scheduling is more difficult. While in s5, s6 and s9, the relationship is a kind of hybrid.

The resulted NU and NRT of different algorithms for the scenarios are presented in Table 7. where Best_PHS is the best result of 1000 PHS generated, and the same to Best_RS and Best_HPS. In each scenario, GA is executed five times with different random seeds. And the average NU and NRT are shown in Table 7. For clarity, we highlight the best results in green, and the second best in yellow. The performance comparison is also illustrated in Fig. 15.

It can be seen from the results that the proposed MGA can always provide the better results in all scenarios. In s4,s8, s9 and s10, MGA can only calculating the second best results. But the results of MGA and the results of PHGA almost the same. In other scenarios, compared other comparison algorithm MGA can calculate best results.

In s4, s8, s9 and s10, the results calculated by PHS are always better than those by RS or HPS due to the geometric period relationship. RS performs better than PHS in s5, s6 and s7, while in other scenarios it is just the opposite. This is why we choose the mixed initial population in MGA.

5. Conclusions

As the GCLs defined in the IEEE 802.1Qbv of TSN must be properly configured before usage, forming an NP-hard flow scheduling problem. To cover both determined and undetermined scenarios, a unified framework is proposed to remodel the problem to utilize the network efficiently while leaving more bandwidth for BE traffic, by optimizing the network utilization and remaining time together. A heuristic algorithm called MGA is proposed to show the effectiveness of the framework. Extensive simulations show that MGA has superiority in various scenarios. In the future, the unified model may be extended to cover more difficult scenarios with multicast flows and not-predefined flow paths.

References

[1] R. Hummen, S. Kehrer, and O. Kleineberg, "Tsn–time sensitive networking," *Hirschmann, USA, WP00027*, 2016.

[2] J. Falk, F. Dürr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for tsn with ilp," in *2018 IEEE 24th Int. Conf. on Embedded and Real-Time Comp. System and Applications (RTCSA)*. IEEE, 2018, pp. 136–146.

[3] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The ieee tsn and ietf detnet standards and related 5g ull research," *IEEE Comm. S&T*, vol. 21, no. 1, pp. 88–145, 2019.

[4] L. Lo Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.

[5] I. S. Association *et al.*, "Ieee standard for local and metropolitan area networks—bridges and bridged networks—amendment 25: Enhancements for scheduled traffic," *Amendment to IEEE Std*, vol. 802, pp. 1–57.

[6] F. Dürr and N. G. Nayak, "No-wait packet scheduling for ieee time-sensitive networks (tsn)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[7] R. Macchiaroli, S. Mole, and S. Riemma, "Modelling and optimization of industrial manufacturing processes subject to no-wait constraints," *International Journal of Production Research*, vol. 37, no. 11, pp. 2585–2607, 1999.

[8] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (tsn)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[9] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2017.

[10] A. M. Kentis, M. S. Berger, and J. Soler, "Effects of port congestion in the gate control list scheduling of time sensitive networks," in *2017 8th International Conference on the Network of the Future (NOF)*, 2017, pp. 138–140.

[11] W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 375–384.

[12] V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–4.

[13] M. Pahlevan and R. Obermaisser, "Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks," in *2018 IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 337–344.

[14] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "Dynamic scheduling and routing for tsn based in-vehicle networks," in *2021*

Table 7
Experiment Results

	Best_PHS		Best_RS		Best_HPS		PHGA		RGA		HPGA		MGA	
	NU	NRT(ns)	NU	NRT(ns)	NU	NRT(ns)	NU	NRT(ns)	NU	NRT(ns)	NU	NRT(ns)	NU	NRT(ns)
s4	0.376378798	15000	0.311565953	9000	0.341874655	14000	0.376414779	-18400	0.321753936	-13600	0.341247997	-31400	0.376354903	-17800
s5	0.229325497	65000	0.232110927	3000	0.249952649	-49000	0.229951325	-5000	0.239072517	-9000	0.250175	2000	0.264749338	11000
s6	0.239508594	-10000	0.252776693	-13000	0.277933594	-28000	0.241069531	-9000	0.258639453	-7000	0.279612891	-29000	0.285685547	-13000
s7	0.15617526	128000	0.173888073	-1000	0.18226151	24000	0.157103385	-12000	0.179291406	14000	0.18397474	12000	0.187988646	85000
s8	0.220098828	1194000	0.220098828	1000	0.220098828	41000	0.220098828	1218000	0.220098828	17000	0.220098828	53000	0.220098828	1198000
s9	0.103112813	1573000	0.103112813	868000	0.103112813	913000	0.103112813	1573000	0.103112813	884000	0.103112813	927000	0.103112813	1570000
s10	0.073113125	2567000	0.073113125	2136000	0.073113125	2115000	0.073113125	2576000	0.073113125	2171000	0.073113125	2108000	0.073113125	2572000

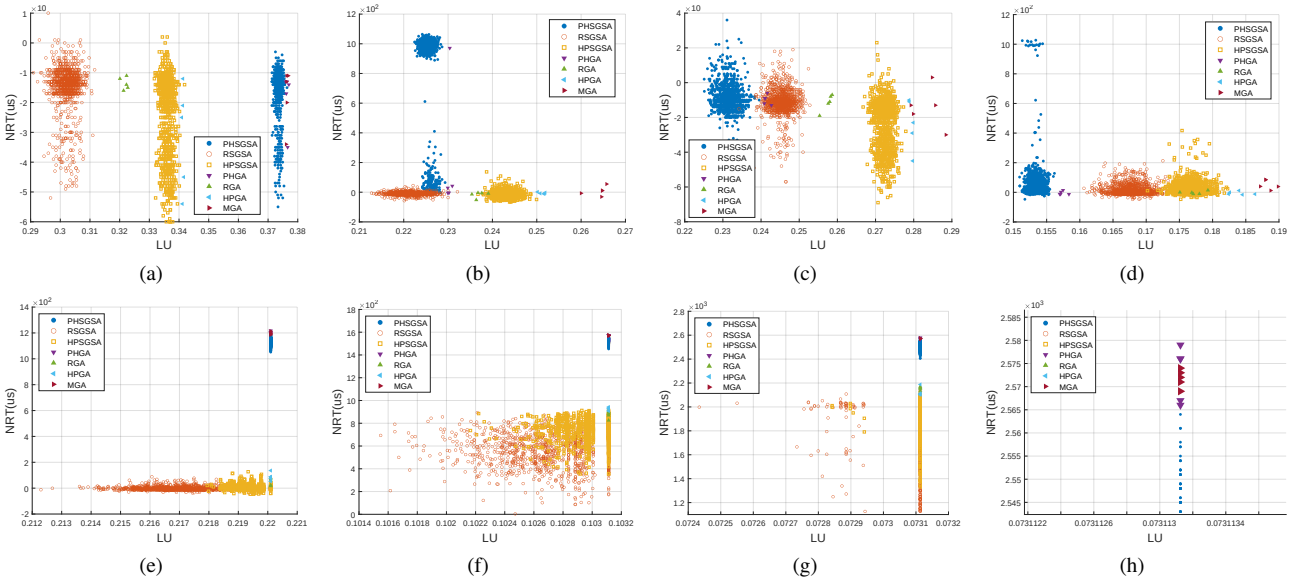


Figure 15: The results of remaining time and network utilization of the algorithms in different scenarios. The results of s4, s5, s6, s7, s8, s9 and s10 are shown in (a)(b) (c) (d) (e) (f) (g) respectively. (h) shows an enlarged view of the some area in (g).

IEEE International Conference on Communications Workshops (ICC Workshops), 2021, pp. 1–6.

[15] A. Arestova, K.-S. J. Hielscher, and R. German, "Design of a hybrid genetic algorithm for time-sensitive networking," in *Measurement, Modelling and Evaluation of Computing Systems*, H. Hermanns, Ed. Cham: Springer International Publishing, 2020, pp. 99–117.

[16] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial ieee 802.1qbv tsn networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4.

[17] M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime reconfiguration of time-sensitive networking (tsn) schedules for fog computing," in *2017 IEEE Fog World Congress (FWC)*. IEEE, 2017, pp. 1–6.

[18] S. S. Craciunas, R. S. Oliver, and T. Ag, "An overview of scheduling mechanisms for time-sensitive networks," *Proceedings of the Real-time summer school LÉcole dÉté Temps Réel (ETR)*, pp. 1551–3203, 2017.

[19] R. Ramaswamy, N. Weng, and T. Wolf, "Characterizing network processing delay," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 3. IEEE, 2004, pp. 1629–1634.

[20] M. D. Johas Teener, A. N. Fredette, C. Boiger, P. Klein, C. Gunther, D. Olsen, and K. Stanton, "Heterogeneous networks for audio and video: Using ieee 802.1 audio video bridging," *Proceedings of the IEEE*, vol. 101, no. 11, pp. 2339–2354, 2013.

[21] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "Avb-aware routing and scheduling of time-triggered traffic for tsn," *IEEE Access*, vol. 6, pp. 75 229–75 243, 2018.

[22] I. Vlašić, M. Đurasević, and D. Jakobović, "Improving genetic algorithm performance by population initialisation with dispatching rules," *Computers & Industrial Engineering*, vol. 137, p. 106030, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219304899

[23] I. Vlašić, M. Đurasević, and D. Jakobović, "Improving genetic algorithm performance by population initialisation with dispatching rules," *Computers & Industrial Engineering*, vol. 137, p. 106030, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219304899

[24] D. Thierens and D. Goldberg, "Convergence models of genetic algorithm selection schemes," in *International conference on parallel problem solving from nature*. Springer, 1994, pp. 119–129.

[25] G. Soremekun, Z. Gürdal, R. Haftka, and L. Watson, "Composite laminate design optimization by genetic algorithm with generalized elitist selection," *Computers & structures*, vol. 79, no. 2, pp. 131–143, 2001.

[26] M. Yamamura, T. Ono, and S. Kobayashi, "Character-preserving genetic algorithms for traveling salesman problem," *JOURNAL-JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE*, vol. 7, pp. 1049–1049, 1992.

[27] M. Boopathi, R. Sujatha, C. S. Kumar, and S. Narasimman, "The mathematics of software testing using genetic algorithm," in *Proceedings of 3rd International Conf. on Reliability, Infocom Technologies and Optimization*. IEEE, 2014, pp. 1–6.

[28] E. N. Gilbert, "Random graphs," *The Annals of Mathematical Statistics*, vol. 30, no. 4, pp. 1141–1144, 1959.